# Competition: An Adaptive Protocol Stack for High-Dependability based on the Population Protocols Paradigm

Dimitrios Amaxilatis
Computer Engineering & Informatics Department
University of Patras
and Computer Technology Institute & Press (CTI)
amaxilat@cti.gr

Ioannis Chatzigiannakis
Department of Computer, Control and Management Engineering
Sapienza University of Rome
and Computer Technology Institute & Press (CTI)
ichatz@dis.uniroma1.it

## Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network monitoring; C.2.4 [**Distributed Systems**]: Distributed applications

## General Terms

Population Protocols, Dependability

*Keywords*

Adaptive Protocols, Algorithm Engineering, Low Power and Lossy Networks

## 1 Introduction

The last decade we witnessed a tremendous progress towards the interconnection of the digital and physical domains, giving rise to the "Internet of Things". An outstanding activity is taking place not only in research but also in commercial areas where ICT is increasingly being embedded into the physical world. Mobile phones, smartphones, NFC, RFID, GPS, and, at a lesser scale, networked sensors, which were placed market-wise until recently as niche products, are now becoming familiar items in our everyday lives. Our homes, our cars, and the things around us are getting fully integrated with the web. The coexistence and cooperation of embedded systems with our social life is unveiling a brand new era of exciting possibilities.

These new systems are characterized by two critical components: (a) dynamicity - the mobility of devices, intermittent connectivity due to lossy links, limited power are some typical factors that create very challenging system models where what was trivially solvable in a static distributed system, is now far from being trivial in this new landscape. (b) robustness - if we expect our societies to depend on future systems, it is imperative that software infrastructures (e.g., cloud of things) will have to be always on, continuously available. The need to make distributed systems more dependable is certainly not new - e.g., Lamport pioneered in the field since the 80s, Tanenbaum made it a crucial design element for the minix system - yet we are by far lacking simple techniques for delivering robust ICT infrastructures.

In this paper we present a protocol stack that is able to adapt to the local environment and coordinate the actions of the IoT devices via simple, local interactions. Our protocol stack is able to detect sudden changes to the structure and operation of the network either due to transient failures (e.g., as a result of the lossy nature of wireless networks) or due to more permanent failures of nodes (e.g., due to limited power resources or due to external forces). Thus the system continues to operate in a set of desired states with maintained, or gracefully degraded or even improved quality of service. Our design adopts the concept of [3] for self-organization that has been widely mentioned in the scope of distributed computing and peer to peer networks.

We implement our solution by following a component-based design using the Wiselib [6]: a code library, that allows implementations to be OS-independent. We totally avoid implementing our algorithm as a monolithic, stand-alone piece of code. The modules can be easily integrated as sub-protocols in other problems such as energy conservation, routing, role assignment, security etc.. We conduct a thorough evaluation using an experimental testbed environment. For all cases, our results indicate that our approach adapts to the external and internal changes. Experimental evidence from real-world deployment indicates that as long as dynamic changes are limited in time and space, they do not affect the rest of the network.

## 2 Adaptive Protocol Stack

An important aspect of our protocol stack is the ability to detect the current topology of the network so that the operation of the devices is properly adjusted to frequent and/or significant topology changes. Beaconing is currently the most effective methodology for neighborhood discovery. With the term beaconing [1] we refer to the exchange of special messages containing information not only for the presence of the node itself but also for its view of the neighborhood.

In contrast to using a network monitoring protocol based on constant periodic beacon exchanges, we chose an adaptive protocol [2] that allows each node to independently modify the beacon broadcast period based on its local perception on the consistency of the network, i.e., the changes to the local neighborhood detected by the node. The protocol uses a

simple rule: as long as a node does not detect any changes on its neighborhood (i.e., *it is consistent*) the beaconing interval period is doubled until a *maximum beaconing interval period* is reached. Therefore, changes to the topology of the network due to failures, interference, or malicious external actions force the nodes affected by the changes to adjust the intensity of the network monitoring process, while unaffected nodes can relax the intensity and achieve lower energy consumption while the network is relatively stable.

The devices use the following indicators for examining the quality of the communication channels are (a) the LQI of the received beacons, (b) the RSSI of the received beacons, (c) the link associativity [5] (the time, in beacons, that nodes are associated, i.e., they retain a connection), (d) the node stability [5] (the average link associativity for all neighboring nodes) and (e) the reverse node stability [2] (indicating the average numbers of consecutive beacons that successfully delivered on a destination).

On top of the network monitoring protocol, our protocol stack provides basic operations for propagation of messages, aggregation of values, symmetry breaking, and detection of critical events using the Population Protocols paradigm [4]. The main benefit of this paradigm is that protocol specifications have no dependence on the number of devices that take part. In other words, since no knowledge about the number of agents is required by the protocols, our protocol stack achieves high scalability for any number of IoT devices. We implement the theoretical model for distributed computation in real IoT environments based on the methodology of [7].

Considering event-driven reporting of critical events, we use the One-Way Epidemic protocol described in [7]. Devices operate in a silent monitoring state and are activated to transmit notification events when the sensor readings match specific criteria. For example, whenever a strong variation in the lighting is detected, the device produces an output '1' bit that signifies that a critical event has occurred. Whenever a device discovers a nearby node (via the networking monitoring protocol) that is not aware of the event (i.e., the local bit is set to '0'), it transmits a short message propagating the critical event. This is the so-called *State Scheduler* [7] which guarantees the acceleration of the epidemic process, delivering the events with at most $O(n)$ interactions (where $n$ is the number of IoT devices).

## 3 Implementation Details

Our protocols are implemented in Wiselib [6] based on C++ and templates, but without virtual inheritance and exceptions. All implemented algorithms are **platform independent** as they can be compiled on a number of different IoT platforms (e.g., TelosB, iSense, ScatterWeb) and **OS independent** as they can be automatically used in systems implemented using C (Contiki), C++ (iSense), and nesC (TinyOS). Interestingly, our code is also compatible with Android and iOS thus they can be executed in SmartPhones and SmartWatches. Clearly, the ability to implement once and run the same code base on such a broad range of IoT devices is a huge advantage that really speeds up development.

Regarding the implementation of the network monitoring sub-protocol, the beacon messages follow the ULA format: the header consists of a $1Byte$ Message id and 2 counters for the number on neighbors and scheduler payloads contained in the message. The list of neighbors is an array of $2Byte$ neighbor MAC addresses in the plain implementation while for the various indicators we add after each address the respective indicator's value as an unsigned $8bit$ integer. Due to hardware limitations of the IoT devices, a single beacon cannot contain an unlimited number of neighbors or piggybacked payloads of very big size. In Wiselib 802.15.4, packets are limited to $116Bytes$ and as a result, it may include a maximum of 37 neighbors. If we need to operate on a larger neighborhood we can use the Wiselib Fragmenting Radio and transmit beacons larger than a single message, allowing us to handle neighborhoods of up to 255 nodes which is the limit of the unsigned integer counter used for the neighbors counter in the message header.

Population protocol *Schedulers* are implemented by piggybacking information on the state of each IoT device on the beacon messages (the "scheduler" payloads). We use the Wiselib's callback mechanism to provide the information as additional payloads attached in each beacon. Thus each node is aware for each neighboring node along with its current state (in terms of the higher-layer population protocols).

## 4 Conclusions

We provide the first population-protocols-based protocol stack implementation for achieving high-dependability in real-world IoT deployments. We evaluate our adaptive protocol stack over a real-world experimental testbed and through large scale simulations. Our results indicate that our protocol stack achieves high-dependability and high-scalability while minimizing battery consumption. Our work suggests many new promising directions in applying the population-protocols paradigm for the development IoT applications.

## 5 References


[1] D. Amaxilatis, I. Chatzigiannakis, S. Dolev, C. Koninis, A. Pyrgelis, and P. Spirakis. Adaptive Hierarchical Network Structures for Wireless Sensor Networks. In *Int. Conference on Ad Hoc Networks*, AdHocNets '11.

[2] D. Amaxilatis, G. Oikonomou, and I. Chatzigiannakis. Adaptive neighbor discovery for mobile and low power wireless sensor networks. In *Proceedings of the 15th ACM International Conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 385–394. ACM, 2012.

[3] E. Anceaume, X. Defago, M. Gradinariu, and M. Roy. Towards a theory of self-organization. *9th International Conference on Principels of Distributed Systems, OPODIS*, pages 146–156, 2005.

[4] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 290–299, New York, NY, USA, 2004. ACM.

[5] A. Bamis, A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52(1):130 – 154, 2008. MSWIM'06.

[6] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, C. Koninis, A. Kröller, and A. Pyrgelis. Wiselib: A generic algorithm library for heterogeneous sensor networks. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, pages 162–177, 2010.

[7] I. Chatzigiannakis, S. Dolev, S. P. Fekete, O. Michail, and P. G. Spirakis. Not all fair probabilistic schedulers are equivalent. In *Principles of Distributed Systems, 13th International Conference, OPODIS 2009, Nîmes, France, December 15-18, 2009. Proceedings*, pages 33–47, 2009.